1. An apparatus in a pipeline microprocessor for avoiding a pipeline stall caused by a load instruction, the microprocessor including execution units, the apparatus comprising:

first dispatch logic, configured to request from a cache data specified by the load instruction, to receive from said cache an indication that said data is not presently available from said cache, and to provide the load instruction and a tag identifying said unavailable data to one of the execution units; and

second dispatch logic in each of the execution units, coupled to said cache, configured to monitor a bus coupling said cache and the execution units to detect a valid match of said tag which indicates said data is now available from said cache, to obtain said data from said cache in response to detecting said valid match of said tag, and to dispatch the load instruction and said data for execution in response to obtaining said data from said cache.

2. The apparatus of claim 1, wherein said first dispatch logic provides the load instruction and said tag to said one of the execution units in response to said indication that said data is not presently available from said cache, rather than stalling the microprocessor pipeline to wait for the data to become available from said cache.

3. The apparatus of claim 1, wherein said second dispatch logic dispatches the load instruction and said data for execution by a functional unit of said one of the execution units.

4. The apparatus of claim 3, wherein said functional unit is configured to move said data into a register maintained by the execution unit.

5. The apparatus of claim 3, wherein said functional unit is configured to perform an operation specified by the load instruction to generate a result.

6. The apparatus of claim 3, wherein said operation comprises an integer arithmetic operation.

7. The apparatus of claim 3, wherein said operation comprises a floating-point arithmetic operation.

8. The apparatus of claim 3, wherein said operation comprises a packed arithmetic operation.

9. The apparatus of claim 8, wherein said operation comprises an MMX packed arithmetic operation.

10. The apparatus of claim 8, wherein said operation comprises an SSE packed arithmetic operation.

11. The apparatus of claim 1, wherein said first dispatch logic is further configured to determine to which of the execution units to provide the load instruction for execution.

12. The apparatus of claim 11, wherein said first dispatch logic determines to which of the execution units to provide the load instruction for execution based on a type of the load instruction.

13. The apparatus of claim 11, wherein said first dispatch logic determines to which of the execution units to provide the load instruction for execution based on a current workload of the execution units.

14. The apparatus of claim 1, further comprising:

an instruction queue, in each of the execution units, configured to store a plurality of the load instruction and said tag received from said first dispatch logic.

15. The apparatus of claim 14, wherein said instruction queue is also configured to store a plurality of said data obtained from said cache.

16. The apparatus of claim 15, wherein said instruction queue is also configured to store a plurality of valid indicators associated with said plurality of said data, each for indicating whether said data is valid.

17. The apparatus of claim 16, wherein if said first dispatch logic receives from said cache said indication that said data is not presently available from said cache, said first dispatch logic also stores a value in said valid indicator to indicate said data is invalid along with providing the load instruction and said tag to said one of the execution units.

18. The apparatus of claim 17, wherein said second dispatch logic is further configured to update said value in said valid indicator to indicate said data is valid in response to obtaining said data from said cache.

19. The apparatus of claim 16, wherein if said first dispatch logic receives from said cache an indication that said data is present in said cache, then said first dispatch logic provides the load instruction and said data from said cache to said one of the execution units and stores a value in said valid indicator to indicate said data is valid.

20. The apparatus of claim 19, wherein said second dispatch logic dispatches the load instruction and said data after said first dispatch logic provides the load instruction and said data to said one of the execution units and stores said value in said valid indicator to indicate said data is valid.

21. The apparatus of claim 1, wherein said first dispatch logic provides the load instruction to said one of the execution units only if the load instruction is determined not to generate an exception.

22. The apparatus of claim 1, wherein said first dispatch logic is further configured to provide non-load instructions and all valid operands to said one of the execution units.

23. The apparatus of claim 1, wherein said tag uniquely identifies said unavailable data.

24. The apparatus of claim 23, wherein said cache manages generation of said tag uniquely identifying said unavailable data to said first dispatch logic.

25. The apparatus of claim 23, wherein said first dispatch logic manages generation of said tag uniquely identifying said unavailable data to said cache.

26. The apparatus of claim 1, wherein said load instruction comprises a microinstruction that loads data from a memory address and performs an arithmetic operation on the data.

27. The apparatus of claim 1, wherein said cache provides to said first dispatch logic said indication that said data is not presently available from said cache if an address of said data misses in said cache.

28. The apparatus of claim 1, wherein said cache provides to said first dispatch logic said indication that said data is not presently available from said cache if an address of said data hits in said cache, but said address in said cache is invalid because said address matches a store operation address in the microprocessor pipeline.

29. The apparatus of claim 28, wherein said pipeline is presently unable to forward said store operation data to said cache.

30. The apparatus of claim 1, wherein said first dispatch logic is located earlier in the microprocessor pipeline than the execution units.

31. The apparatus of claim 1, wherein said second dispatch logic obtains said data from said cache via said bus.

32. The apparatus of claim 1, wherein said load instruction comprises an instruction that instructs the microprocessor to read said data from a memory location specified by a memory address into the microprocessor.

33. The apparatus of claim 1, wherein a computer program product comprising a computer usable medium having computer readable program code causes the apparatus, wherein said computer program product is for use with a computing device.

34. The apparatus of claim 1, wherein a computer data signal embodied in a transmission medium comprising computer-readable program code provides the apparatus.

35. A pipeline microprocessor, comprising:

a data cache, coupled to a bus;

a plurality of execution units, each coupled to said data cache by said bus, each having an instruction queue and dispatch logic for dispatching instructions from said queue to a functional unit of the execution unit for execution thereof after said instruction queue indicates all operands of the instructions are valid; and

an instruction dispatcher, coupled to said plurality of execution units, configured to provide to said instruction queues of said plurality of execution units instructions and operands thereof and an indication of whether said operands are valid, wherein if an instruction operand is unavailable from said data cache, said instruction dispatcher provides said instruction to said instruction queue with an indication that said unavailable operand is invalid, rather than stalling subsequent instructions in the pipeline to wait for said operand from said data cache.

36. The microprocessor of claim 35, wherein said instruction dispatcher is further configured to provide to said instruction queue a tag uniquely identifying said unavailable operand.

37. The microprocessor of claim 36, wherein said dispatch logic is further configured to subsequently obtain said unavailable operand from said data cache via said bus in response to detecting a match of said tag previously provided by said instruction dispatcher with a tag transmitted on said bus by said data cache along with said unavailable operand.

38. The microprocessor of claim 37, wherein said dispatch logic is further configured to subsequently dispatch said instruction and said previously unavailable operand to said functional unit for execution in response to obtaining said previously unavailable operand from said data cache via said bus.

39. The microprocessor of claim 35, wherein said plurality of execution units comprise at least an integer execution unit, a floating-point execution unit, an MMX execution unit, and an SSE execution unit.

40. The microprocessor of claim 35, wherein said instruction comprises a load instruction.

41. A method for performing distributed load instruction dispatching in a pipeline microprocessor having a cache, a central instruction dispatcher and a plurality of execution units each having auxiliary instruction dispatching logic, the method comprising:

receiving, by the central instruction dispatcher, from the cache an indication that operand data specified by a load instruction is unavailable;

providing, by the central instruction dispatcher, the load instruction to one of the plurality of execution units and indicating that the data is unavailable; and

obtaining, by the auxiliary instruction dispatching logic in the one of the plurality of execution units, the operand data from the cache subsequent to said indicating to the one of the plurality of execution units that the data is unavailable.

42. The method of claim 41, further comprising:

dispatching for execution, by the auxiliary instruction dispatching logic in the one of the plurality of execution units, the instruction and the obtained operand data to a functional unit in the one of the plurality of execution units subsequent to said obtaining.

43. The method of claim 41, further comprising:

determining whether the load instruction is an exception-generating instruction; and

said providing the load instruction to the one of the plurality of execution units only if the load instruction is not an exception-generating instruction.

44. The method of claim 41, further comprising:

providing, by the central instruction dispatcher, a tag uniquely identifying the unavailable data to the one of the plurality of execution units prior to said obtaining, by the auxiliary instruction dispatching logic in the one of the plurality of execution units, the operand data from the cache.

45. The method of claim 44, further comprising:

transmitting, by the cache, the tag uniquely identifying the unavailable data on a bus subsequent to said providing, by the central instruction dispatcher, the tag uniquely identifying the unavailable data to the one of the plurality of execution units.

46. The method of claim 45, further comprising:

detecting, by the auxiliary instruction dispatching logic in the one of the plurality of execution units, a match of the tag uniquely identifying the unavailable data on the bus in response to said transmitting; and

said obtaining, by the auxiliary instruction dispatching logic in the one of the plurality of execution units, the operand data from the cache in response to said detecting.

47. The method of claim 41, further comprising:

obtaining, by the data cache, the operand data subsequent to said indicating, by the central instruction dispatcher, to the one of the plurality of execution units that the data is unavailable.

48. The method of claim 41, wherein said providing, by the central dispatch logic, the load instruction to the one of the plurality of execution units comprises writing the load instruction into an instruction queue of the one of the plurality of execution units.

49. A method for performing detached operand load operations in a microprocessor, the method comprising:

decoding a first instruction, the first instruction specifying an operand specified by a memory address;

providing the first instruction to a first execution unit without the operand in response to the memory address missing in the cache memory and indicating to the first execution unit that the operand is invalid;

obtaining, by the first execution unit, the operand from the cache memory subsequent to said providing the first instruction;

executing, by the first execution unit, the instruction in response to said obtaining;

decoding a second instruction after said decoding the first instruction; and

providing the second instruction to a second execution unit prior to said obtaining the operand from the cache memory.

50. The method of claim 49, further comprising:

providing the instruction to the execution unit along with the operand if the memory address hits in the cache memory and indicating to the execution unit that the operand is valid.

51. The method of claim 49, further comprising:

providing a tag uniquely identifying the missing operand substantially concurrently with said providing the first instruction to the first execution unit without the operand; and

detecting, by the execution unit, a match of the tag with a tag transmitted by the cache memory along with the operand, wherein said obtaining, by the first execution unit, the operand from the cache memory is performed in response to said detecting.

52. A computer data signal embodied in a transmission medium, comprising:

computer-readable program code for providing n apparatus in a pipeline microprocessor for avoiding a pipeline stall caused by a load instruction, the microprocessor including execution units, said program code comprising:

first program code for providing first dispatch logic, configured to request from a cache data specified by the load instruction, to receive from said cache an indication that said data is not presently available from said cache, and to provide the load instruction and a tag identifying said unavailable data for execution to one of the execution units; and

second program code for providing second dispatch logic in each of the execution units, coupled to said cache, configured to monitor a bus coupling said cache and the execution units to detect a valid match of said tag which indicates said data is now available from said cache, to obtain said data from said cache in response to detecting said valid match of said tag, and to dispatch the load instruction and said data in response to obtaining said data from said cache.